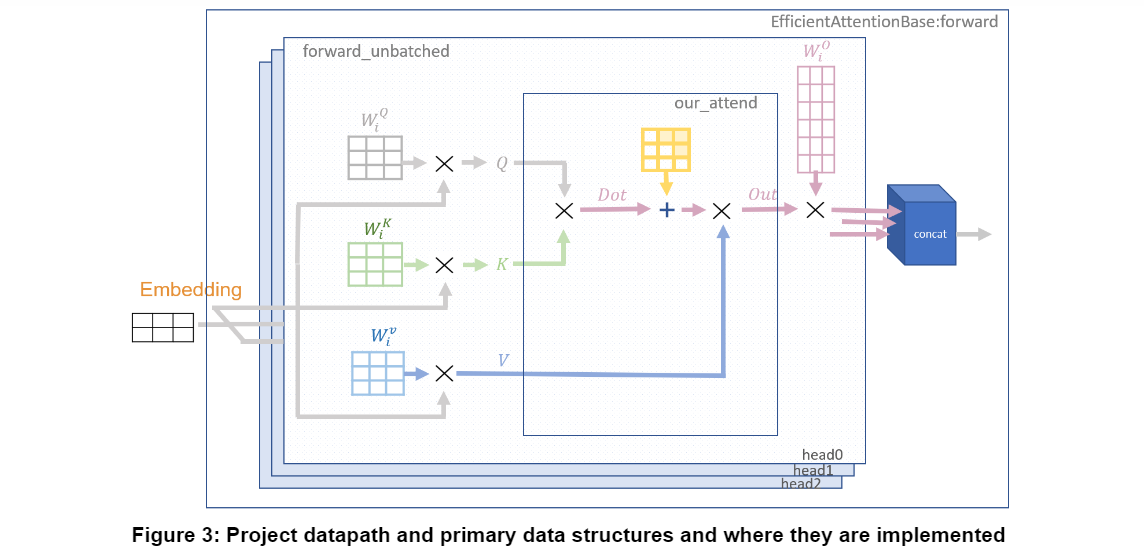
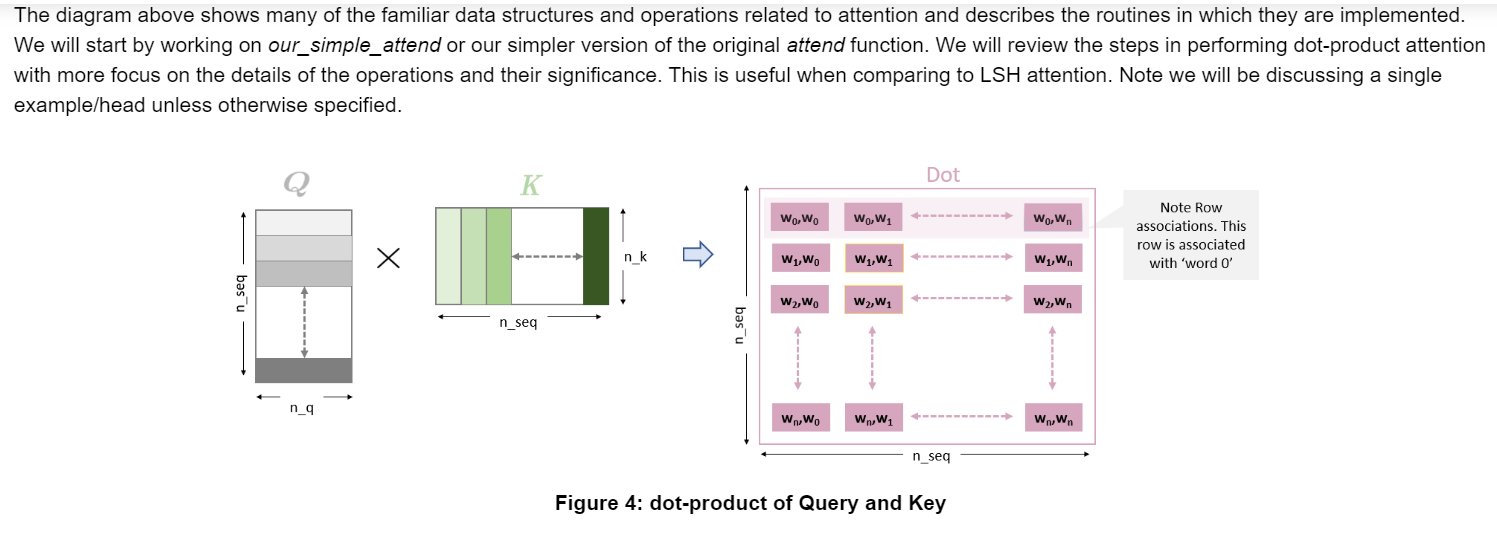
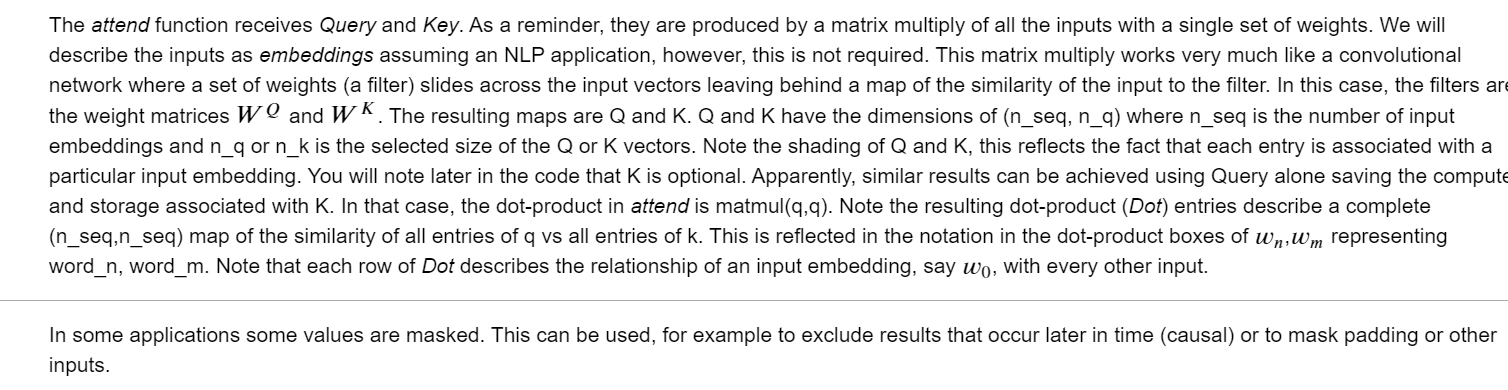
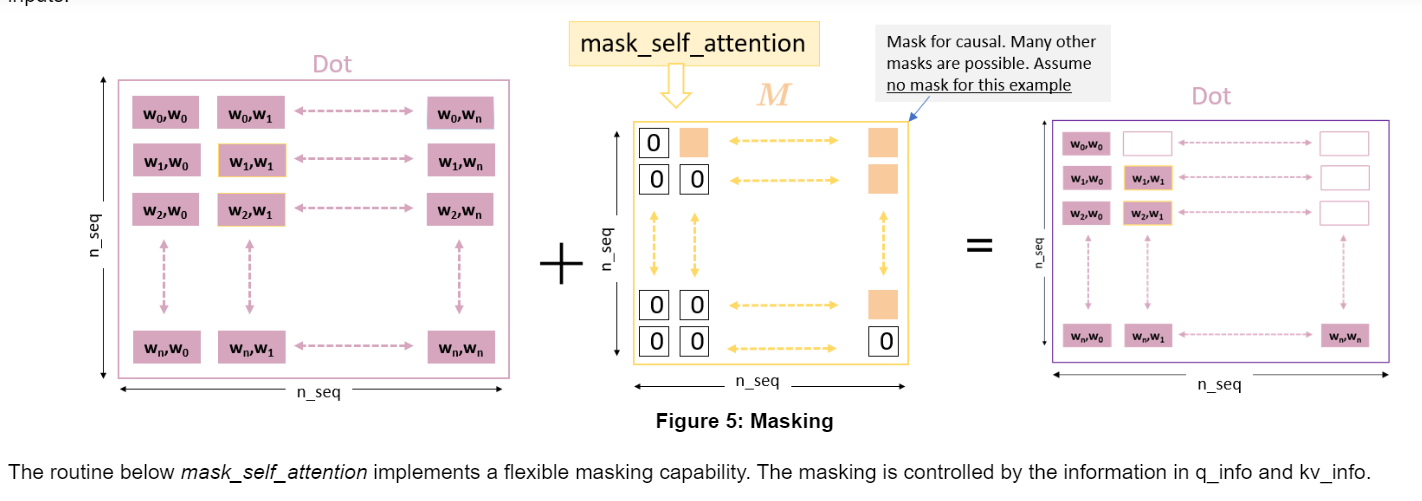
***Reformer Transformers***

Full Dot Product Self-Attention



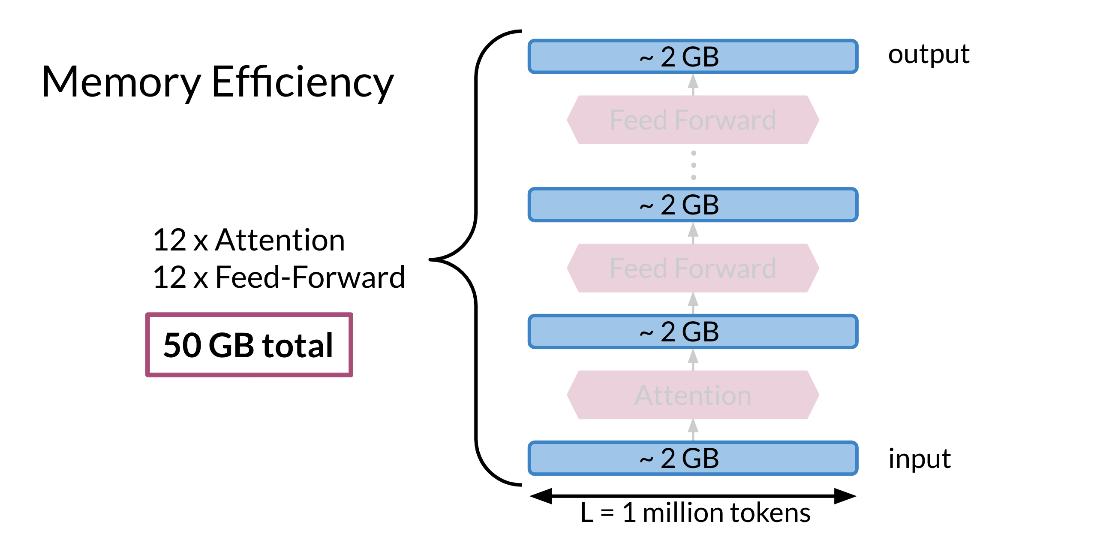






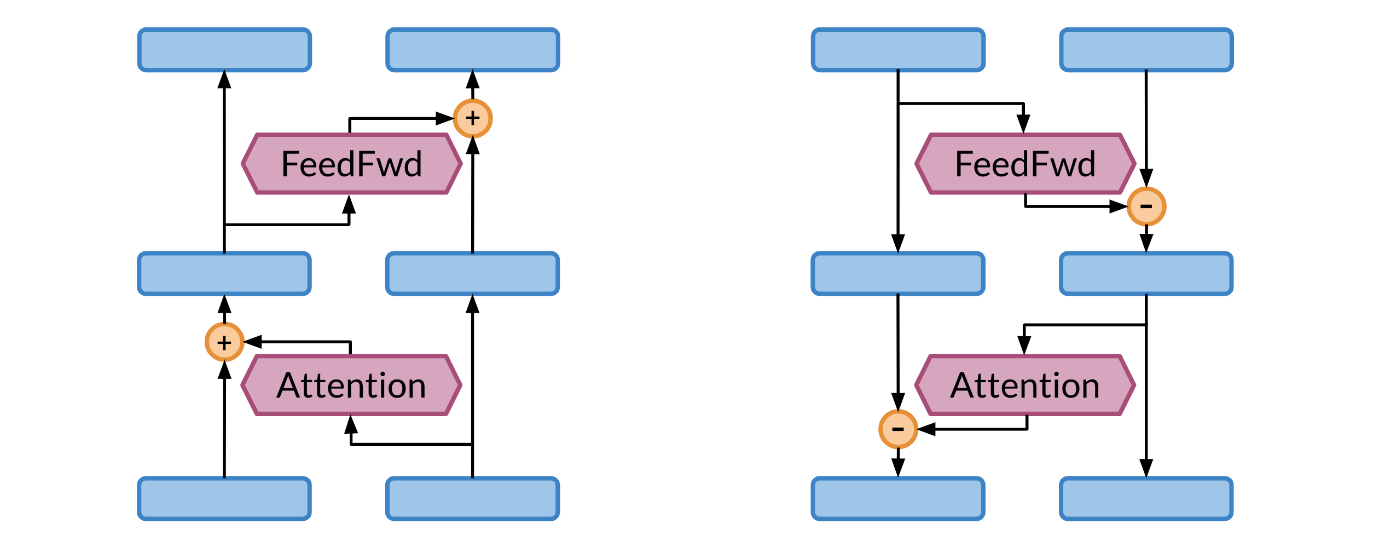
Motivation for Reversible Layers: Memory!

Every time you run a forward propagation, you need to compute the back propagation to update the weights. The biggest issue with doing this is that you have to store the weights to be able to compute the back-prop. With these very large models, that could be a lot of memory.

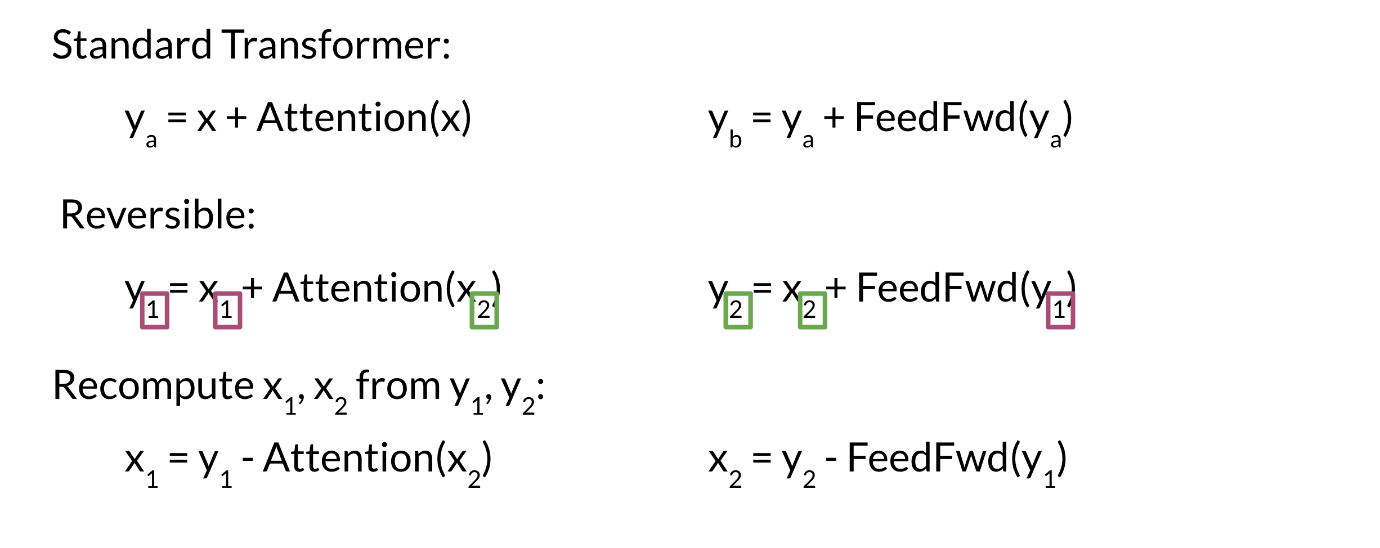


For example in the model above it requires 2GB to compute the Attention and 2GB for the feed forward. You have 12 layers for attention and 12 layers for the feedforward. That is equal to 12 \* 2 + 12\*2 + 2 (for the input) = 50 GB. That is a lot of memory. In the next video you will learn how to solve such problems.

Reversible residual layers allow you to reconstruct the forward layer from the end of the network. Usually you have two similar branches in the network that you use to compute the network.



In the left picture, you have the forward propagation. One side of the network is used as input and the other is used for the attention. In the left side, the same thing is happening but in the opposite direction.



X2 and X1 are copies of each other.

Take a few minutes and try to understand the equations above. You basically make use of the two branches of the network. When coming back for the back propagation, you only need the y's to compute x\_2*x*2​ and then you can use x\_2*x*2​ along with y\_1*y*1​ to compute x\_1*x*1​. Pretty neat! Now you don't have to store the weights, because you can just compute them from scratch. This image shows you a visualization of what is happening.

